Systems Considerations for V&V and UQ at Exascale

Dan Gunter, Lavanya Ramakrishnan

Computational Research Division

Lawrence Berkeley National Laboratory

Challenge

"The uncertainty quantification effort will lead to ensembles of simulations with as many as one million members, requiring new techniques for understanding the data sets."

p. 16, "Scientific Discovery at the Exascale: Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization"

Can current systems do this?

No.

Most HPC systems can't even run the jobs

- Batch queues can't handle the truth
  - have issues with mere *hundreds* of jobs
  - jobs-per-user in queue O(10)
- Clouds aren't any better

Q: How many grad students and jobs does it take to crash NERSC Carver cloud queue?

A: 1

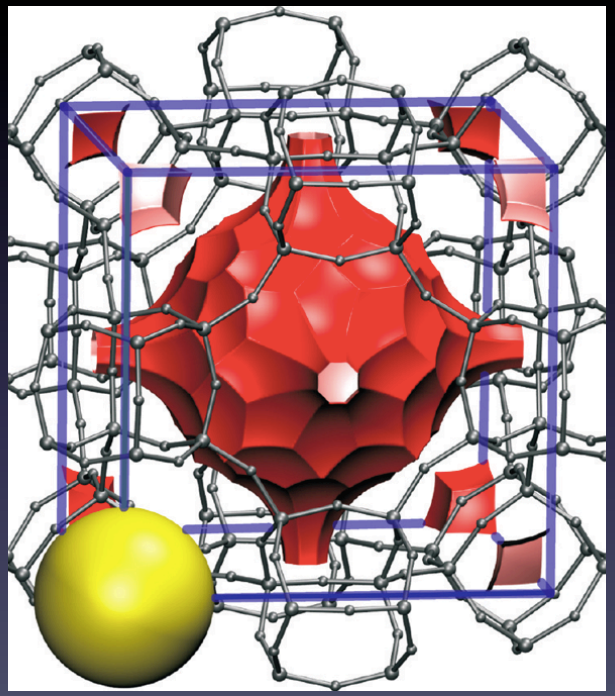
No grad students were harmed in the making of this slide. Actually a picture of my collaborator, Lavanya Ramakrishnan

If they could run the jobs, they wouldn't help with the data

- Offered abstraction: files and directories
  - hard to search
  - where to put metadata?
  - fragile, brittle, etc. etc.
- Result: Roll-your-own framework

Broader context

- Not just UQ:
  - Materials Genome
  - Carbon Capture
  - Meta-genomics
  - Earthquake simulations
  - *… shout out your favorite!*



Abstraction: Code Ensembles

- **Code Ensembles**
  - A large number of loosely coupled tasks, each with their own internal parallelism
- What % have we talked about that fits that definition?
- How many of the *middleware* aspects are being re-invented each time?

Gaps in job management tools

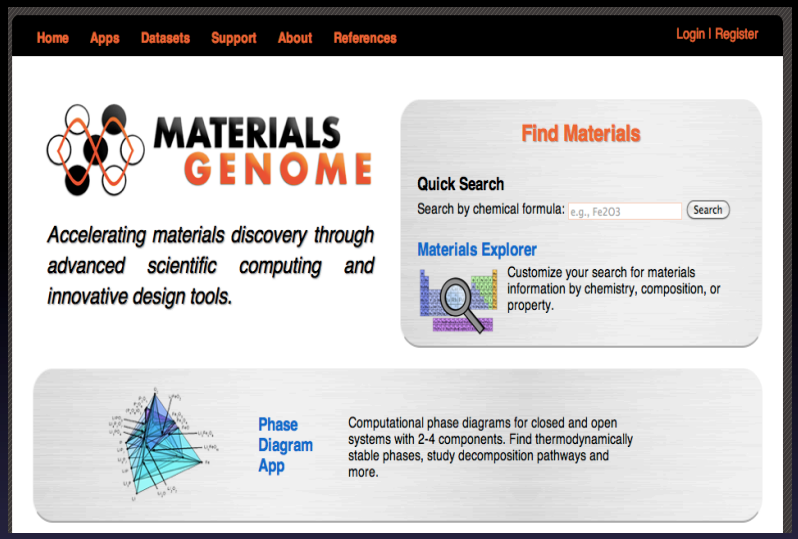
- UQ-specific tools
  - Multi-level parallelism, but assumes single batch queue
  - Some monitoring and fault tolerance
  - Data management is file-based
- Workflow tools (Pegasus, Taverna, Kepler, etc.)
  - Mature tools from distributed/grid computing
  - Focused on simply acquiring resources
  - Do not deal well with dynamic elements, HPC batch queues

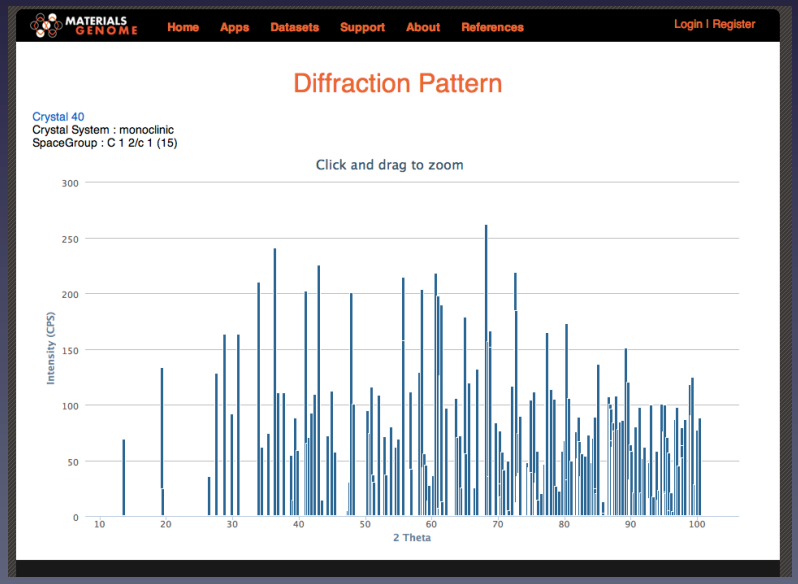
Gaps in provenance and data management tools

- Have some formats for scientific data
  - HDF5, netCDF, etc.
- But for metadata, and semi-structured data in general, it's all application-specific
- Few general tools for manual or automated provenance
- External access to data? File transfers. No DropBox ☹

Materials Genome Project

- Run material structure calculations on ~125,000 known crystals
- Store results in a database
- Provide web interface for researchers to explore database and run "apps" to calculate, e.g., diffraction patterns and phase diagrams
- Expected official release: October, 2011
- Collaboration between LBNL and MIT





MG Automation

(1) Database is initialized with workflow inputs, tasks, and dependencies

(2) Engines register and pull tasks

(3) Engines send back:
- Results
- Logs
- *New tasks*

Engine

Hopper

Engine

Carver

Engine

Lawrencium

Engine

& etc.

A word about the Database

- Project at MIT used PostgreSQL (RDBMS)
  - E/R diagram looked like a car accident
  - Common queries 1-2 pages of SQL
- Switched to MongoDB "NoSQL" DB
  - Schemaless, but fast and scalable
  - Easily used for coordination and data storage
    - linking these 2 has benefits!
  - So, so much easier..

Exascale: Ready or not

System Parameter	2011	"2018" Swim Lane 1	"2018" Swim Lane 2	Factor Change
System Peak	2 Pf/s	1 Ef/s		500
Power	6 MW	≤20 MW		3
System Memory	0.3 PB	32–64 PB		100–200
Total Concurrency	225K	1B×10	1B×100	40,000–400,000
Node Performance	125 GF	1 TF	10 TF	8–80
Node Concurrency	12	1,000	10,000	83–830
Network BW	1.5 GB/s	100 GB/s	1000 GB/s	66–660
System Size (nodes)	18700	1,000,000	100,000	50–500
I/O Capacity	15 PB	300–1000 PB		20–67
I/O BW	0.2 TB/s	20–60 TB/s		10–30

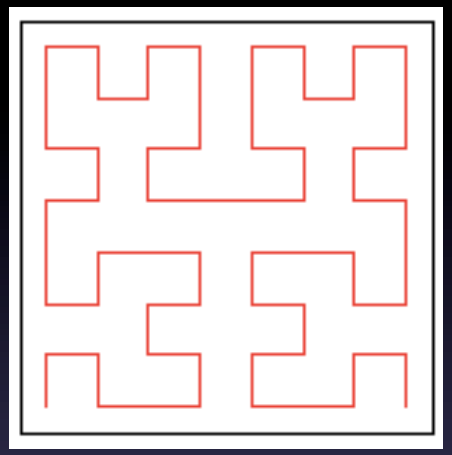
Source: Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization. Houston, TX, Feburary 2011.

I/O Challenge

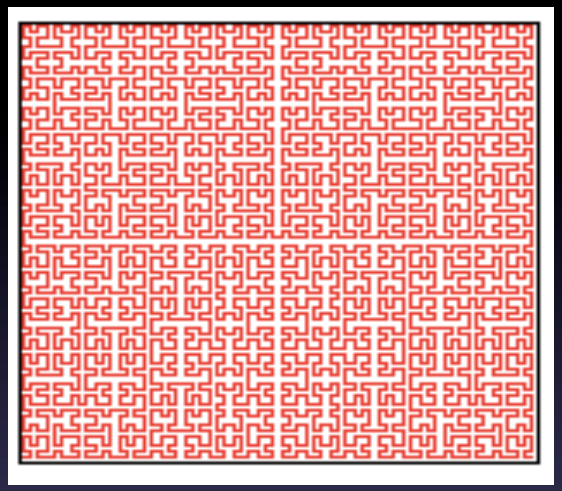
- Concurrency is going bananas
- Aggregate I/O bandwidth is not keeping up
- Disk/disk data movement is expensive (time and power), need *in situ* analysis
- **Does V&V/UQ make this worse or better?**

Fault tolerance challenge

MTBF =~ 0.5 month



MTBF =~ 1 day? hour?



- Cannot do traditional checkpointing because of the I/O bottleneck
  - Need to integrate checkpointing with the algorithm
- **Can UQ ensembles optimize for fault-tolerance?**

Data challenge

We need to be able to *store*, *find*, *compare, and share* data at a large scale.

Current file/directory hierarchy is not good enough.

Can V&V and UQ practices guide metadata needed?

Requirements

Provide

- Fault-tolerance
- I/O efficiency (*in situ*)
- Data management
- Dynamic task management
  - monitoring
- UQ-specific support

Be

- Scalable for large #jobs and data products
- Flexible, to handle new data types easily
- Lightweight and low-overhead on HPC systems
- Usable by mere mortals

Do UQ folks really want to spend their time dealing with this?

I hope not.

Path forward

- Develop common tools:
  - Abstract out the middleware aspects (fault-tolerance, provenance, data-management, etc.) from current UQ frameworks
- Let UQ *drive* dynamic aspects of new approach
- Hopefully will encourage even more cross-dissemination of techniques

UQ Study Group at LBNL

- Organized a weekly study group at LBNL to share ideas about current practice, applications
- Anyone from this group is invited to speak!
  - probably can't pay travel costs
  - visiting the Bay Area is its own reward

Fin.